Infrastructure Automation using Terraform

Create a S3 Bucket – Implementing Variables

accenture

Table of Contents

This Activity demonstrates the implementation of variables while creating S3 Bucket in AWS using Terraform. We can assign values to arguments using variables within Terraform. We have numerous ways to define a variable. So, if we assign multiple values to the same variable then terraform will consider the value which has highest precedence. The order of precedence from low to high is as follows:

> ➢ Variable block Definition – "default" argument value
> ➢ terraform.tfvars
> ➢ terraform.tfvars.json
> ➢ *.auto.tfvars or *.auto.tfvars.json
> ➢ Command-Line (-var or --var-file)
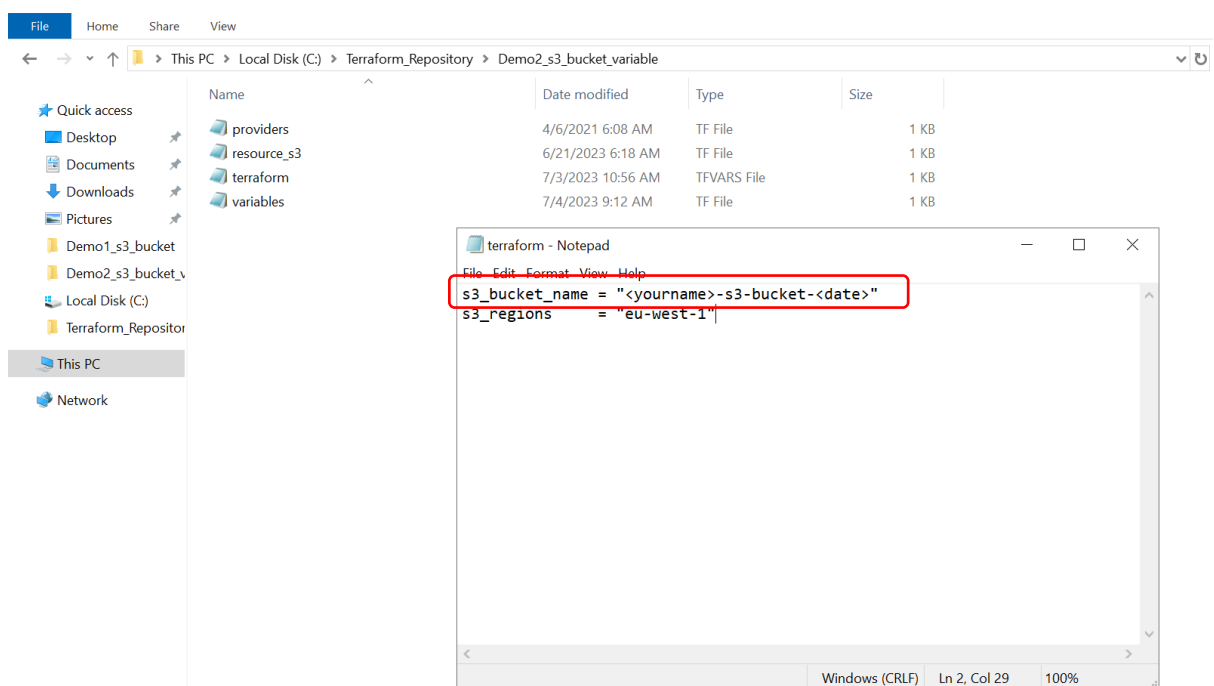
Prerequisite:

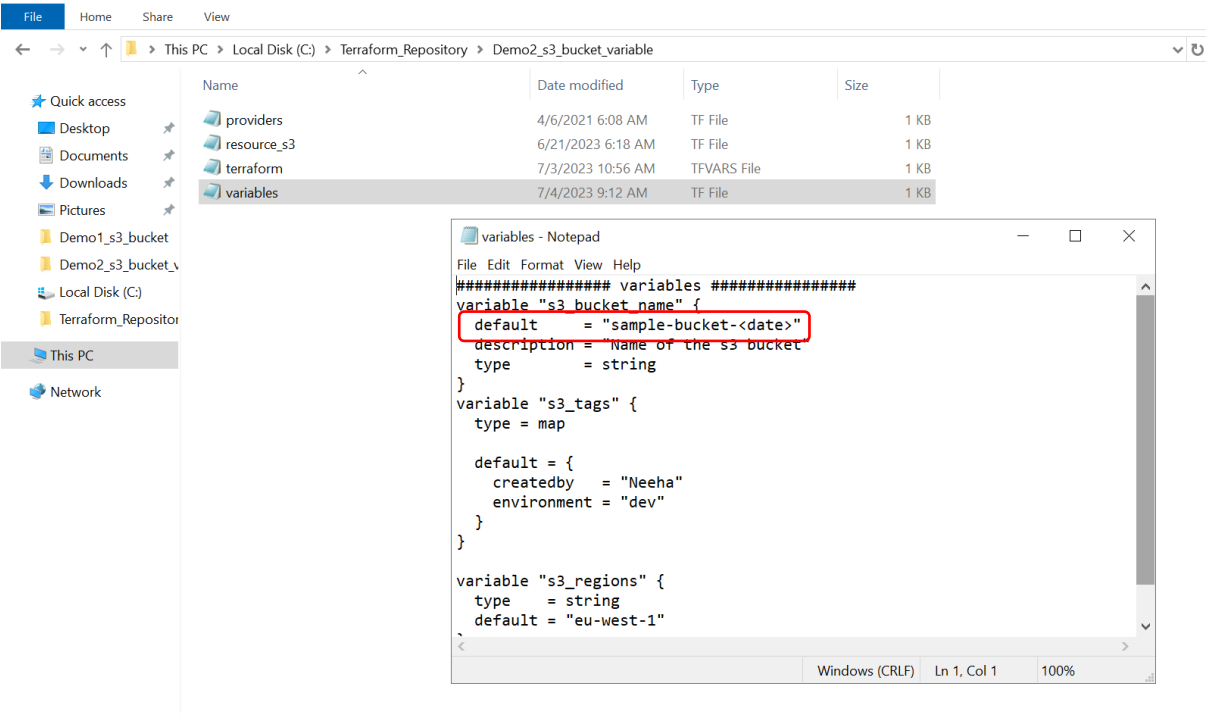1) Download the zip file shared by the trainer and extract it.

Walkthrough:

1. Initializing Terraform Directory
2. Creating S3 Bucket – Implementing Variables
3. Destroying S3 Bucket

Part 1: Initializing Terraform Directory

1	Open the extracted folder and navigate to ".tf" files. Open "variables.tf" , "terraform.tfvars" and update the "s3_bucket_name" argument in both the files.
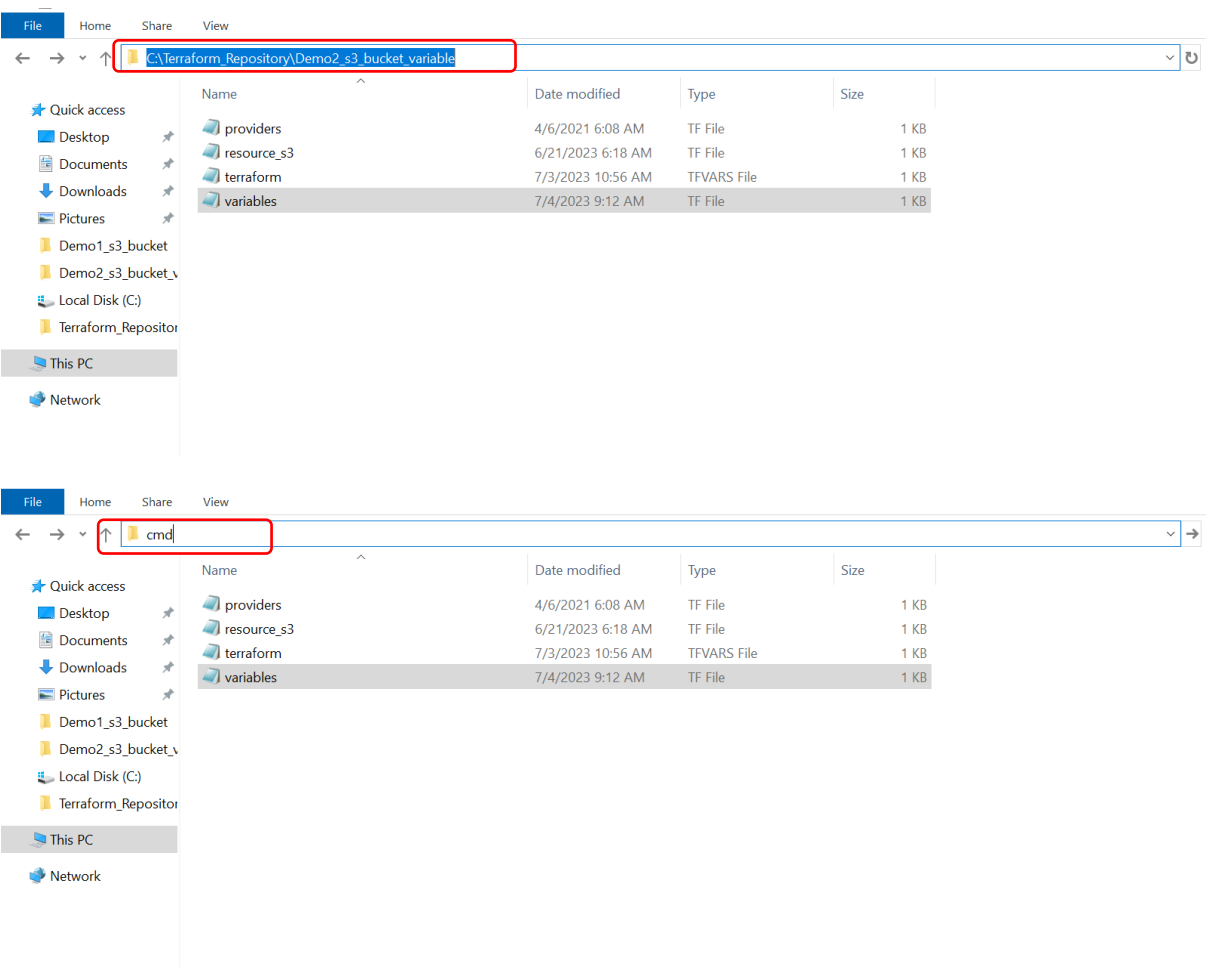
2	Click on the Address bar and type cmd. Press Enter (It will open a command prompt from that location).

| 3 | Execute below command to initialize the current directory as Terraform directory which enables us to run terraform commands to manage Infrastructure.

Command :

```
C:\Terraform_Repository\Demo2_s3_bucket_variable>terraform init
```

Result :

```
C:\Terraform_Repository\Demo2_s3_bucket_variable>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.9.0...
- Installed hashicorp/aws v5.9.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

C:\Terraform_Repository\Demo2_s3_bucket_variable>
``` |
|---|---|
| 4 | Next execute below command to validate syntax and configuration of terraform configuration files. If everything is proper, it will return a success message otherwise it will display the errors.

Command :

```
C:\Terraform_Repository\Demo2_s3_bucket_variable>terraform validate
``` |

	Result : ``` C:\Terraform_Repository\Demo2_s3_bucket_variable>terraform validate Success! The configuration is valid. C:\Terraform_Repository\Demo2_s3_bucket_variable>_ ```

5	Next run below command and observe the output. The output contains information depicting all the changes which will happen in the AWS cloud. It is like dry-run to ensure whatever we are trying to do using terraform commands is what we want.
	
	Command :
	
	``
	C:\Terraform_Repository\Demo2_s3_bucket_variable>terraform plan -out "s3_variable.tfplan"
	``
	
	Result :
	
	``
	C:\Terraform_Repository\Demo2_s3_bucket_variable>terraform plan -out "s3_variable.tfplan"
	
	Terraform used the selected providers to generate the following execution plan. Resource actions are indicated
	with the following symbols:
	  + create
	
	Terraform will perform the following actions:
	
	  # aws_s3_bucket.mybucket will be created
	  + resource "aws_s3_bucket" "mybucket" {
	      + acceleration_status = (known after apply)
	      + acl = (known after apply)
	      + arn = (known after apply)
	      + bucket = "neeha-s3-bucket-2707"
	      + bucket_domain_name = (known after apply)
	      + bucket_prefix = (known after apply)
	      + bucket_regional_domain_name = (known after apply)
	      + force_destroy = false
	      + hosted_zone_id = (known after apply)
	      + id = (known after apply)
	      + object_lock_enabled = (known after apply)
	      + policy = (known after apply)
	      + region = (known after apply)
	      + request_payer = (known after apply)
	      + tags = {
	``
	
	NOTE : After executing above command , you will notice that S3 bucket is getting created with the name which you have provided in "terraform.tfvars" file, the value define in "variable.tf" file is getting ignored or overridden.

Part 2: Creating S3 Bucket – Implementing Variables

1	For creating a S3 Bucket , execute below command and observe the actions performed by the command. **Command :** ``` C:\Terraform_Repository\Demo2_s3_bucket_variable>terraform apply "s3_variable.tfplan" ``` **Result :** ``` C:\Terraform_Repository\Demo2_s3_bucket_variable>terraform apply "s3_variable.tfplan" aws_s3_bucket.mybucket: Creating... aws_s3_bucket.mybucket: Creation complete after 4s [id=neeha-s3-bucket-2707] Apply complete! Resources: 1 added, 0 changed, 0 destroyed. C:\Terraform_Repository\Demo2_s3_bucket_variable>_ ```

Part 3: Destroying S3 Bucket

1	Execute below command to destroy the S3 Bucket which we have created in previous step. After you execute below command, it will show you what changes will be done and before doing those changes it will ask for your approval. So, if you want to proceed with destroying S3 Bucket, provide "yes".

Command:

```
C:\Terraform_Repository\Demo2_s3_bucket_variable>terraform destroy
```

Result:

```
C:\Terraform_Repository\Demo2_s3_bucket_variable>terraform destroy
aws_s3_bucket.mybucket: Refreshing state... [id=neeha-s3-bucket-2707]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated
with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_s3_bucket.mybucket will be destroyed
  - resource "aws_s3_bucket" "mybucket" {
      - arn                         = "arn:aws:s3:::neeha-s3-bucket-2707" -> null
      - bucket                      = "neeha-s3-bucket-2707" -> null
      - bucket_domain_name          = "neeha-s3-bucket-2707.s3.amazonaws.com" -> null
      - bucket_regional_domain_name = "neeha-s3-bucket-2707.s3.eu-west-1.amazonaws.com" -> null
      - force_destroy               = false -> null
      - hosted_zone_id              = "Z1BKCTXD74EZPE" -> null
      - id                          = "neeha-s3-bucket-2707" -> null
      - object_lock_enabled         = false -> null
      - region                      = "eu-west-1" -> null
      - request_payer               = "BucketOwner" -> null
      - tags                        = {
          - "env" = "dev"
        } -> null
      - tags_all                    = {
          - "env" = "dev"
```

```
Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: _
```

```
Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_s3_bucket.mybucket: Destroying... [id=neeha-s3-bucket-2707]
aws_s3_bucket.mybucket: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.

C:\Terraform_Repository\Demo2_s3_bucket_variable>_
```
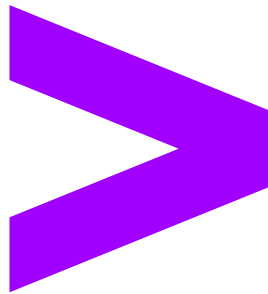
Infrastructure Automation using Terraform – Lab Guide